НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2018р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки6.050103 “ Програмна інженерія “

на тему: Web-система моніторингу довкілля за аналізом набору зображень

Виконав: студент 4_курсу, групи__ТВ-з51__

_____Дормідонтов Валентин Сергійович_____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____ст.в., Мірошниченко І.В._____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2018р.

ЗАВДАННЯ
на дипломну роботу студенту
Дормідонтову Валентину Сергійовичу
(прізвище, ім’я, по батькові)

1.Тема роботи Web-система моніторингу довкілля за аналізом набору зображень
керівник роботи Мірошниченко Іван Володимирович, ст.в.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 2018р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи: Web-система моніторингу довкілля за аналізом набору зображень
4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)проаналізувати задачу створення web-системи моніторингу довкілля за набором зображень на прикладі завантажуємого фото та розпізнавання на ньому об`єктів та їх пошкоджень.

5. Перелік ілюстративного матеріалу: схеми архітектури додатку, знімки екранних форм, діаграма прецедентів системи, діаграма структури системи, зразки розробленого інтерфейсу додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-6	Шалденко О.В. к.т.н., ст.викл.		

7. Дата видачі завдання ”6”_листопада_____2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Програмна реалізація системи		
5.	Оформлення пояснювальної записки		
6.	**1** Захист програмного продукту		
7.	**2** Передзахист		
8.	Захист		

Студент _____ Дормідонтов В. С.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Мірошниченко І.В.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дана дипломна робота присвячена розробці web-системи по моніторингу довкілля за набором зображень на прикладі завантаженого фото та розпізнавання на ньому об`єктів та їх пошкоджень.

Створена програма здатна ідентифікувати об’єкти на зображенні за допомогою набору тренувальних даних та побудованої моделі. Також розпізнавати пошкодження на виявленому об’єкті та виводити результат у вигляді обробленого зображення з маркованим дефекотом. Забезпечує інтуїтивний інтерфейс користувача і можливість додавання тренувальних наборів даних та об’єктів для їх подальшого моніторингу.

Для забезпечення виявлення пошкоджень було реалізовано кілька сучасних алгоритмів комп’ютерного зору. Кінцевий результат роботи у програмі являє собою марковане зображення з виявленими дефектами що базується на вхідному зображенні.

Ключові слова: нейронна мережа, пошкодження, визначення об’єків, сегментація.

Обсяг звіту становить 64 сторінок, міститься 18 ілюстрацій, 2 додатки. Загалом опрацьовано 13 джерел.

ABSTRACT

This thesis is devoted to the development of a web-system for monitoring the environment by a set of images, for example, a loaded photo and the recognition of objects on it and their damage.

A created program is capable of identifying objects in an image using a set of training data and a built-in model. Also recognize the damage on the detected object and output the result in the form of a processed image with a marked defect. Provides an intuitive user interface and the ability to add training sets of data and objects for their further monitoring.

Several modern computer-based algorithms were implemented to ensure the detection of damage. The end result of the program is a marked image with defects found based on the input image.

Keywords: Neural Network, Damages, Object Definition, Segmentation.

Scope of the report is 64 pages contain 18 illustrations, 2 annexes. Generally 13 sources have been processed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЗНМ	–	Згорткова нейронна мережа
UI	–	Інтерфейс користувача (User interface)
DFD	–	Діаграма даних (Data Flow Diagram)

ВСТУП

За останні кілька десятків років технологія комп’ютерного зору стає більш популярною та вдосконалюється. Перші спроби розробки та впровадження таких технологій беруть початок у середині 20-го сторіччя. Але на той час не було таких потужних систем що давали змогу ефективного розвитку та впровадження в сфери життя та праці людей. Тим не менш, на даний момент людство має більш доступні та потужні для розвитку ресурси. Як зі сторони програмного забезпечення, так із сторони технічного обладнання обчислювальних систем. Також має місце наявність розробок штучного інтелекту що полегшує розробку та швидкість алгоритмів аналізу зображень.

Аналіз та цифрова обробка зображень використовується в основному для їх покращення, наприклад у мобільних пристроях таких як мобільний телефон, камера, тощо. Тим не менш, алгоритми комп’ютерного зору дають можливість отримати частини зображення та їх атрибути для подальшої роботи та аналізу.

Одною з актуальних задач на даний момент для комп’ютерного зору є розпізнавання образів на зображенні для отримання інформації. Велика частина даної області присвячена застосуванням згорткових нейронних мереж.

Застосування комп’ютерного зору є дуже різноманітним, воно охоплює системи безпеки, контроль виготовлення об’єктів, системи візуального контролю та управління, контроль промислових об’єктів.

Успішне застосування машинного зору не потребує фундаментальних та поглиблених знань і навичок в різних областях комп’ютерних наук, таких як бази даних, мережеві системи та інше.

Задачі розпізнавання об’єктів, а також їх пошкоджень, можуть розглядатися на прикладі аналізу різноманітних споруд. Наприклад, це можуть бути промислові об’єкти, пам’ятки архітектури, важкодоступні споруди що віддалені від наземних

пунктів або які знаходяться в несприятливих для людини умовах. Також, має місце автоматизація нагляду за станом споруд та конструкцій. Тож потрібна система, яка буде допомагати користувачу, обробляти велику кількість даних та зосереджуватися на важливих деталях.

Для отримання цих результатів мною була розроблена web-система, яка дозволяє вирішити поставлені задачі.

1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ WEB-СИСТЕМИ МОНІТОРИНГУ ДОВКІЛЛЯ ЗА НАБОРОМ ЗОБРАЖЕНЬ

Метою даної дипломної роботи є створення web-системи по моніторингу довкілля за набором зображень, а саме виявлення об’єктів та їх пошкоджень.

При розробленні відповідного забезпечення потрібно розв’язати наступні завдання:

1. Провести попереднє завантаження фото на сервер.
2. Створення набору даних для ідентифікації об’єкту.
3. Ідентифікувати об’єкт на зображенні за набором даних.
4. Провести зображення об’єкту та виявити його пошкодження.
5. Обробити зображення виділивши об’єкт та його пошкодження.
6. Вивести результат обробки користувачу.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

З кожним роком з’являються все нові і нові інструменти моніторингу довкілля засновані на різних принципах та ідеях, але вони є важкими для впровадження, громіздкими та потребують додаткових професійних навичок. Тому проблема розробки такої системи залишається досі актуальною

Саме тому було запропоновано до розробки саме систему моніторингу за довкіллям, яка дозволила б скоротити час та людські ресурси на рутинну роботу та автоматизувати моніторинг об’єктів та їх пошкоджень зосереджуючи увагу лише на важливих даних минуючи обробку людиною нерелевантних.

2.1. Аналіз предметної області: сегментація зображення

Сегментація зображень - це завдання розбиття цифрового зображення на одне або декілька областей, що представляють інтерес. Це фундаментальна проблема в області комп'ютерного зору, яка вирішується багатьма різними способами, кожен з яких має свої переваги та недоліки.[1]

У різних галузях часто виникає потреба отримання сегменованих образів. Наприклад, в медицині, де існує інтерес до отримання цифрових 3D моделей органів для подальшого аналізу. Також сегментація застосовується в розпізнаванні осіб, машинному зору тощо.

2.2. Аналіз існуючих алгоритмів сегментації зображення

Сегментація зображень має декілька популярних алгоритмічних підходів для розв’язування задач — Watershed, Floodfill, Grabcut.

Алгоритм Watershed працює з зображенням як з функцією від двох змінних f =I(x,y), де x,y — координати пікселів.

Рисунок 2.1 – Приклад роботи алгоритму Watershed

Значенням функції може бути інтенсивність або модуль градієнта. Для найбільшого контрасту можна взяти градієнт від зображення. Якщо по осі OZ відкладати абсолютне значення градієнта, то в місцях перепаду інтенсивності утворюються хребти, а в однорідних регіонах - рівнини. Після знаходження мінімумів функції f, йде процес заповнення "водою", який починається з глобального мінімуму. Як тільки рівень води досягає значення чергового локального мінімуму, починається його заповнення водою. Коли два регіони починають зливатися, будується перегородка, щоб запобігти об'єднанню областей. Вода продовжить підніматися до тих пір, поки регіони не відділятися тільки штучно збудованими перегородками як на рисунку 2.2.

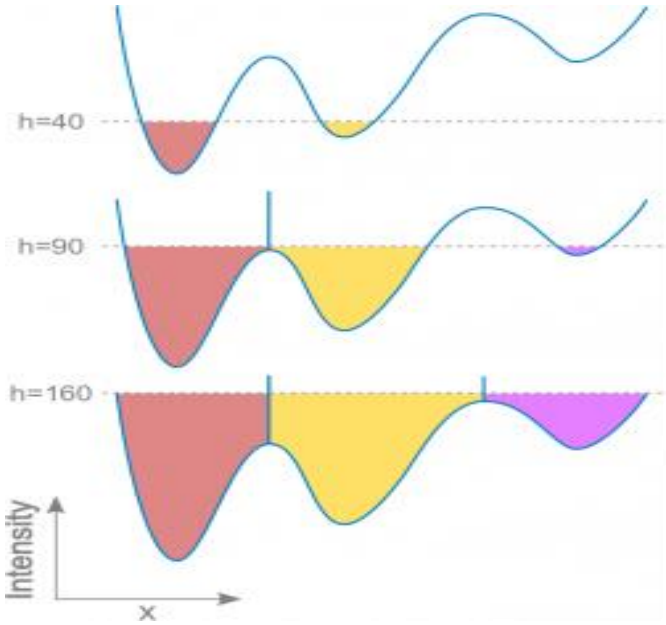


Рисунок 2.2 – Приклад заповнення

Такий алгоритм може бути корисним, якщо на зображенні невелике число локальних мінімумів, в разі ж їх великої кількості виникає надлишкове розбиття на сегменти.[2] Щоб позбутися від надлишку дрібних деталей, можна задати області, які будуть прив'язані до найближчих мінімумам. Перегородка буде будуватися тільки в тому випадку, якщо відбувається об'єднання двох регіонів з маркерами, в іншому випадку буде відбуватися злиття цих сегментів. Такий підхід прибирає ефект надлишкової сегментації, але вимагає попередньої обробки зображення для виділення маркерів, які можна позначити інтерактивно на зображенні.

За допомогою FloodFill (заливка або метод «заповнення») можна виділити однорідні за кольором регіони. Для цього потрібно вибрати початковий піксель і задати інтервал зміни кольору сусідніх пікселів щодо вихідного. Інтервал може бути і несиметричним. Алгоритм буде об'єднувати пікселі в один сегмент (заливаючи їх одним кольором), якщо вони потрапляють в зазначений діапазон. На виході буде сегмент, залитий певним кольором, і його площа в пікселях.

Такий алгоритм може бути корисний для заливки області зі слабкими перепадами кольору однорідним фоном. Одним з варіантів використання FloodFill може бути виявлення пошкоджених країв об'єкта. Наприклад, якщо, заливаючи

однорідні області певним кольором, алгоритм заповнить і сусідні регіони, то значить порушена цілісність кордону між цими областями. Нижче на зображенні можна помітити, що цілісність кордонів заливаються областей зберігається:

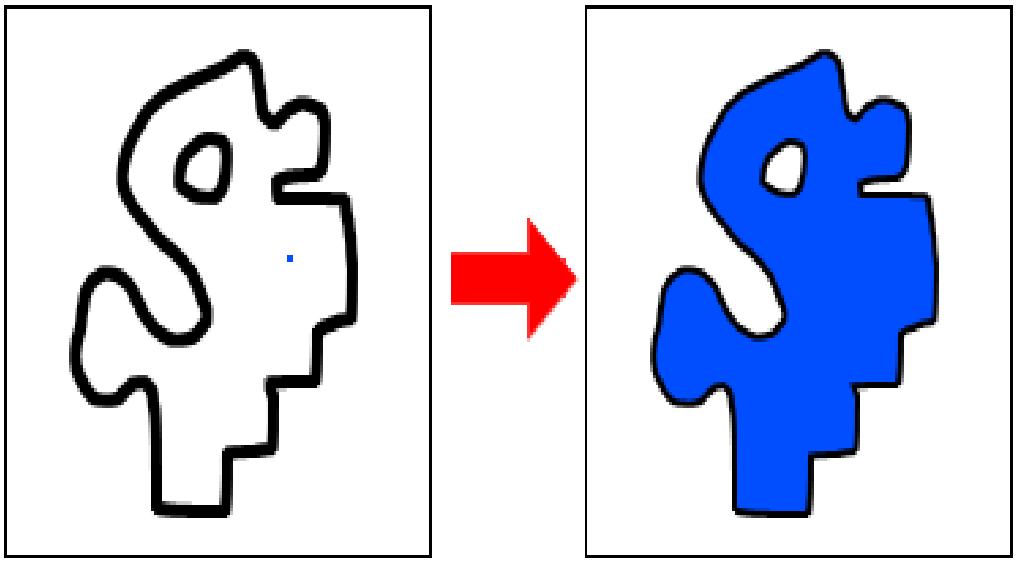


Рисунок 2.3 – Приклад роботи FloodFill

GrabCut - це інтерактивний алгоритм виділення об'єкта, розроблявся як більш зручна альтернатива магнітному ласо (щоб виділити об'єкт, користувачеві потрібно обвести його контур за допомогою миші). Для роботи алгоритму досить укласти об'єкт разом з частиною фону в прямокутник (grab).[3] Сегментування об'єкту відбудеться автоматично (cut) як показано на рисунку 2.4.

Рисунок 2.4 – Приклад роботи GrabCut

2.3. Висновки

Із зробленого аналізу можна зробити висновки про складність реалізації системи моніторингу за набором зображень. Це може бути причиною громіздкості або відсутності відкритого і доступного рішення, а от відсутність попиту, можна виключити як можливий фактор, адже моніторингом довкілля фахівці займаються кожного дня. Це можуть бути пам’ятки архітектури, помислові об’єкти, тощо.

Також, цілком можливо, що деякі рішення цієї задачі існують, але лише у вигляді закритих технологій, розроблених у межах великих підприємств, що мають велику інфраструктуру та обсяги промисловості.

Розробка відкритого універсального програмного рішення задачі моніторингу та виявлення пошкоджень на об’єктах може мати велику цінність для декількох індустрій, включаючи індустрію малих підприємств та державних організацій охорони довкілля.

3 АНАЛІЗ ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ

Найкращі результати в області розпізнавання об'єктів показала Convolutional Neural Network або згорткова нейронна мережа, яка є логічним розвитком ідей таких архітектур нейромереж як когнітрону і неокогнітрона. Успіх обумовлений можливістю обліку двовимірної топології зображення, на відміну від багатошарового персептрона.[4]

Згорткові нейронні мережі забезпечують часткову стійкість до змін масштабу, зсувів, поворотам, зміні ракурсу і інших спотворень. Згорткові нейронні мережі об'єднують три архітектурних ідеї, для забезпечення інваріантності до зміни масштабу, повороту зрушення і просторовим спотворень:

1. Локальні рецепторні поля (забезпечують локальну двовимірну зв'язність нейронів);

2. Загальні вагові коефіцієнти синапсів (забезпечують детектування деяких рис в будь-якому місці зображення і зменшують загальне число вагових коефіцієнтів);

3. Ієрархічна організація з просторовими підвибірки.

На даний момент згорткова нейронна мережа і її модифікації вважаються кращими по точності і швидкості алгоритмами знаходження об'єктів на зображеннях.

3.1 Структура згорткової нейромережі

ЗНМ складається з різних видів шарів: згорткові шари, шари підвиборки і шари «звичайної» нейронної мережі відповідно до рисунку 3.1:

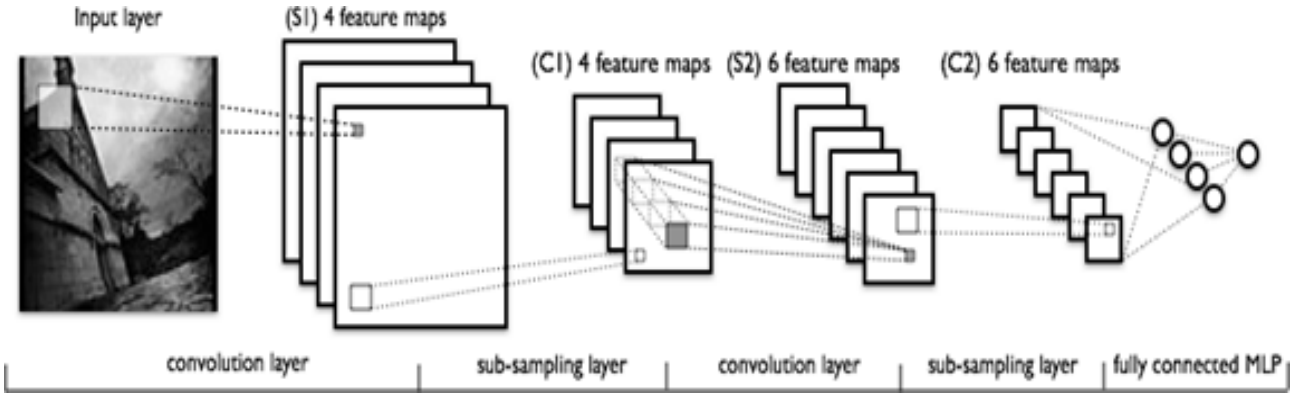


Рисунок 3.1 –Топологія ЗНМ

Перші два типи шарів, чергуючись між собою, формують вхідний вектор ознак для багатошарового персептрона.

Свою назву згорткова мережа отримала за назвою операції - згортка, суть якої буде описана далі.

Згорткові мережі є вдалою серединою між біологічно правдоподібними мережами і звичайним багатошаровим персептроном. На сьогоднішній день кращі результати в розпізнаванні зображень отримують з їх допомогою. В середньому точність розпізнавання таких мереж перевершує звичайні ІНМ на 10-15%. ЗНМ - це ключова технологія глибкого навчання (Deep Learning).[5]

Основною причиною успіху ЗНМ стала концепція загальних ваг. Незважаючи на великий розмір, ці мережі мають невелику кількість параметрів, що налаштовуються в порівнянні з їх предком - неокогнітроном. Є варіанти ЗНС схожі на неокогнітрон. В таких мережах відбувається часткова відмова від пов'язаних ваг, але алгоритм навчання залишаютсься тими же і ґрунтуються на зворотному поширенні помилки.[6] ЗНС можуть швидко працювати на послідовній машині і швидко навчатися за рахунок чистого розпаралелювання процесу згортки по кожній карті, а також зворотної згортки при поширенні помилки по мережі.

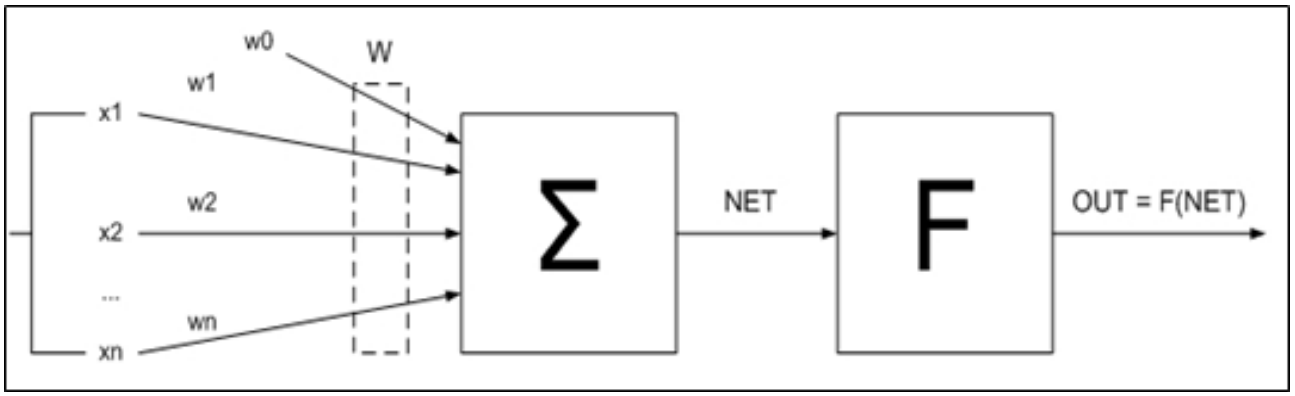


Рисунок 3.2 –Модель нейрона

На рисунку 3.2 зображена модель нейрона, де w_i - вага нейрона, x_i — допоміжний параметр або зсув, n — кількість зв’язків що входять до нейрону.[7]

3.2 Топологія згорткової нейромережі

Визначення топології мережі орієнтується на поставлені завдання, дані з наукових статей і власний експериментальний досвід.

Можна виділити наступні етапи що впливають на вибір топології:

1. Визначити поставлену задачу використовуючи нейромережі (класифікація, прогнозування, модифікація);

2. Визначити обмеження в розв'язуваної задачі (швидкість, точність відповіді);

3. Визначити вхідні (тип: зображення, звук) і вихідних дані (кількість класів).

Розв'язувана моєї нейромережею завдання - класифікація зображень, конкретно промислових об’єктів. Накладається обмеження на мережу - це точність розпізнавання не менше 70%. Загальна топологія мережі відповідно до рисунку 3.3.

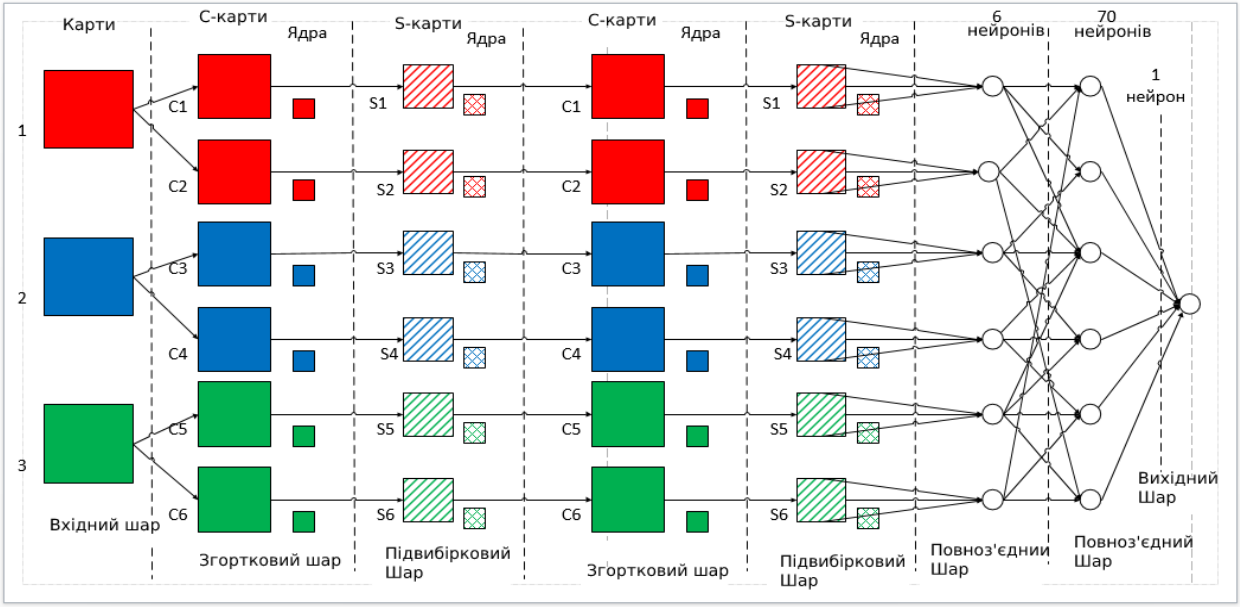


Рисунок 3.3 –Топологія нейромережі

3.3 Вхідний шар

Вхідні дані являють собою кольорові зображення типу JPEG, розміру 600x600 пікселів. Якщо розмір буде занадто великий, то обчислювальна складність підвищиться, відповідно обмеження на швидкість відповіді будуть порушені, визначення розміру в даній задачі вирішується методом підбору. Якщо вибрати розмір занадто маленький, то мережа не зможе виявити ключові ознаки осіб. Кожне зображення розбивається на 3 канали: червоний, синій, зелений. Таким чином виходить 3 зображення розміру 600x600 пікселів.

Вхідний шар враховує двовимірну топологію зображень і складається з декількох карт (матриць), карта може бути одна, в тому випадку, якщо зображення представлено в відтінках сірого, інакше їх 3, де кожна карта відповідає зображенню з конкретним каналом (червоним, синім і зеленим) .

Вхідні дані кожного конкретного значення пікселя нормалізуються в діапазон від 0 до 1, за формулою:

$$f(p, min, max) = \frac{p - min}{\max - min} \tag{3.1}$$

Де f — функція нормалізації, p — значення кольору пікселя від 0 до 255, min — мінімальне значення пікселя 0, max — максимальне значення пікселя 255.[8]

3.4 Згортковий шар

Згортковий шар являє собою набір карт (матриць), у кожної карти є синаптичне ядро (фільтр).[9]

Кількість карт визначається вимогами до задачі, якщо взяти велику кількість карт, то підвищиться якість розпізнавання, але збільшиться обчислювальна складність. Виходячи з аналізу наукових статей, в більшості випадків пропонується брати співвідношення один до двох, тобто кожна карта попереднього шару (наприклад, у першого згорткового шару, попереднім є вхідний) пов'язана з двома картами згорткового шару, відповідає зображенню з конкретним каналом (червоним, синім і зеленим).

3.5 Висновки до розділу

У розділі розглянуті основні методи та алгоритми для обробки та сегментації зображення з застосуванням згорткових нейронних мереж.

Це завдання низького рівня часто є першим кроком у складних системах комп’ютерного зору де точність інтерпретації кінцевої сцени сильно залежить від якості наданої сегментації.

4 ЗАСОБИ ДЛЯ ВИКОНАННЯ ЗАДАЧІ

Для виконання поставленої задачі були використані бібліотек TensorFlow та фреймворку Keras на основі архітектури MobileNet а також фреймвоку Django а мові програмування Python.

4.1 Бібліотека Tensorflow

Бібліотека (в програмуванні) - це файл або набір файлів, що містять підпрограми, функції, які використовуються для розробки програмного забезпечення. Різні мови програмування мають свій набір бібліотек.[10]

Виділяють:

Стандартні бібліотеки - поставляються з мовою програмування.

Призначені для користувача бібліотеки - створюються користувачами або програмістами.

Бібліотека TensorFlow помітно спрощує вбудовування в додатки елементів і функцій штучного інтелекту, призначених для розпізнавання мови, комп'ютерного зору або обробки природної мови.

TensorFlow не єдина бібліотека глибинного навчання, але, як і пошуковий механізм Google, вона вважається кращою в своєму класі. Альтернативами є програмне забезпечення Torch, створене швейцарськими дослідниками, і розробка Каліфорнійського університету в Берклі Caffe, остання версія якої, Caffe2, була спроектована за участю Facebook.

Бібліотека підходить для широкого сімейства технік машинного навчання, а не тільки для створення нейронних мереж. Лінійна алгебра та інші принципи, на яких

базується фреймворк, добре видно зовні. На додаток до основної функціональності машинного навчання, TensorFlow також включає власну систему логування, власний інтерактивний візуалізатор логів і навіть потужну архітектуру з надання даних. Модель виконання TensorFlow відрізняється від scikit-learn мови Python і від більшості інструментів в R.

4.2 Фреймворк Keras

Keras – це фреймворк, який являє собою надбудову над бібліотеками для створення моделей нейронних мереж. Він має добре спроектоване API, що дозволяє будувати модель шляхом складання блоків високого рівня.[11] Цей фреймворк є написанимна Python та може використовуватися в якості надбудови над такими бібліотеками, як TensorFlow, CNTK чи Theano. Keras було створено для швидкої розробки моделей машинного навчання. Основні властивості фреймворка, що визначають зручність використання у розробці:

—Дозволяє легко і швидко створювати прототипи (за допомогою зручності, модульності та розширюваності).

—Підтримує як згорткові мережі, так і рекурентні мережі, а також комбінації обох.

—Для обчислень може використовувати як процесор, так і графічну відеокарту.

4.3 Фреймворк Django

Django – це безкоштовний і вільний фреймворк для веб-додатків, написаний на Python. Архітектура Django нагадує шаблон «Модель-Вид-Контролер» (MVC), однак в якості «контролеру» в Django фактично виступає «вид», а в якості «виду» - «шаблон». Таким чином, розробники Django модель MVC назвали MVT («Модель-Вид-Шаблон»).

Даний фреймворк володіє наступними перевагами:

1) швидкість: Django був розроблений, щоб допомогти розробникам створити додаток настільки швидко, наскільки це можливо. Це включає в себе формування ідеї, розробку і випуск проекту, де Django економить час і ресурси на кожному з цих етапів. Таким чином, його можна назвати ідеальним рішенням для розробників, для яких питання дедлайну варто в пріоритет;

2) повна комплектація: Django працює з десятками додаткових функцій, які помітно допомагають з аутентифікацією користувача, картами сайту, адмініструванням вмісту, RSS тощо. Дані аспекти допомагають здійснити кожен етап веб-розробки;

3) безпека: працюючи в Django, користувач отримує захист від помилок, пов'язаних з безпекою, які ставлять під загрозу проект. Це можуть бути ін'єкції SQL, крос-сайт підробки, і крос-сайтовий скриптинг. Для ефективного використання логінів і паролів, система користувальницької аутентифікації є ключем;

4) масштабованість: фреймворк Django найкращим чином підходить для роботи з високими трафіками. Тому велика кількість сайтів використовують Django для задоволення вимог, пов'язаних з трафіком;

5) різнобічність: менеджмент контенту, наукові обчислювальні платформи, навіть великі організації - з усім цим можна ефективно справлятися за допомогою Django .

4.4 Архітектура MobileNet

Архітектута Mobilenet на даний момент є найефективнішою в порівнянні з іншими арітектурами нейронних мереж. Серед переваг можна відмітити легковісність, гнучкість та оптимізацію. Її навіть можна використовувати на мобільних пристроях та комп'ютерах які не мають великих потужностей.

Звичайна згортка представляє собою фільтр $D_k * D_k * C_{in}$, де D_k — це розмір ядра згортки, а C_{in} — кількість каналів на вході. Загальна обчислювальна важкість згорткового шару має $D_k * D_k * C_{in} * D_f * D_f * C_{out}$, де D_f — висота та ширина шару (ми вважаємо що просторові розміри вхідгного та вихідного тензора співпадають), а C_{out} — кількікість каналів на виході.[12]

Ідея depthwise separable convolution полягає в тому, щоб розкласти подібний шар на depthwise-згортку, яка вдає із себе поканально фільтр, і 1x1-згортку (також відому як pointwise convolution). Сумарна кількість операцій для застосування такого шару дорівнює $(D_k * D_k + C_{out}) * C_{in} * D_f * D_f$.

Згорткова частина що нас цікавить складається з одного звичайного згорткового шару шару з 3x3 згорткою на початку і тринадцяти блоків, зображених справа на малюнку 4.1, з поступово зростаючою кількістю фільтрів і спадаючою просторовою розмірністю тензора. Зліва намальований блок звичайної згорткової мережі, а праворуч - базовий блок MobileNet.

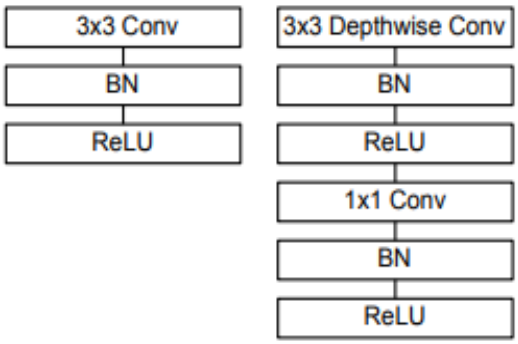


Рисунок 4.1 –Різниця між звичайною згортковою мережею та MobileNet

Особливістю даної архітектури є відсутність max pooling-шарів. Замість них для зниження просторової розмірності використовується згортка з параметром stride, рівним 2. Двома гіперпараметрамі архітектури MobileNet є α (множник ширини) і ρ (множник глибини або множник дозволу). Множник ширини відповідає за кількість каналів в кожному шарі. Наприклад, α = 1 дає нам архітектуру, описану в вище, а α = 0.25 - архітектуру зі зменшеним в чотири рази числом каналів на виході кожного блоку.

Множник дозволу відповідає за просторові розміри вхідних тензорів. Наприклад, ρ = 0.5 означає, що висота і ширина feature map, яка подається на вхід кожного шару буде зменшена вдвічі. Обидва параметра дозволяють варіювати розміри мережі: зменшуючи α і ρ, ми знижуємо точність розпізнавання, але в той же час збільшуємо швидкість роботи і зменшуємо споживану пам'ять.[13]

4.5 Висновки до розділу

У цьому розділі було розглянуто засоби для виконання задачі такі як TensorFlow, Keras, Djabngo та архітектуру MobileNet.

Вибір даних засобів ґрунтується на стабільності, гнучкості та спроможності масштабуватися. Також засоби не потребують значних потужностей та об’єму ресурсів для ефективної роботи.

5 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Web-система призначена для двох акторів – користувача, що бажає перевірити стан об’єкту та адміністратора — що може додавати камери спостереження та створювати або змінювати користувачів та групи доступу..

Для того, щоб мати можливість користуватися системою користувач повинен увійти у систему.

5.1 Структура програми

Програмне забезпечення для розв’язання поставленої задачі спроектовано з дотриманням парадигм об’єктно-орієнтованого програмування та паттерну MVC.

Під час розробки програмного продукту була реалізована триланкова архітектура[13] задля гнучкої роботи web-системи, рисунок 5.1 являє собою кліент-серверний додаток, використовуючи який користувач спілкується з сервером через браузер. Результат запитів виводиться на HTML сторінках.

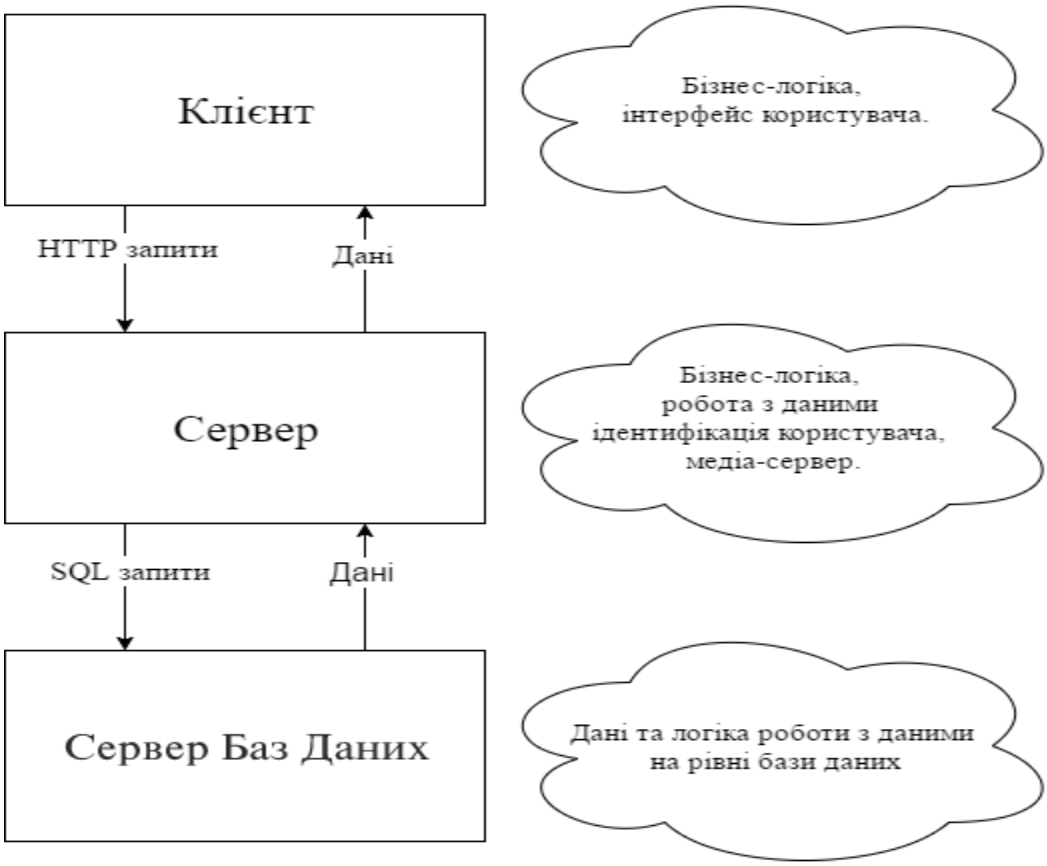


Рисунок 5.1–Архітектура web-системи

Спочатку користувач повинен пройти увійти у систему, потім завантажити фото у набір даних на сервер для подальшого тренування моделі. Дані, що були отримані під час визначення об`єктів на зображенні, записуються у базу даних.

Після тренування моделі користувач може добати веб-камеру, що містить об’єкт, та матиме змогу перевіряти його стан.

5.2 Опис розроблених алгоритмів

Для розробки даного програмного продукту по сегментації та обробці зображення у вигляді web-системи були використані такі технології як феймворки Angular6 та AngularMaterial для відображення інтерфейса користувача в повному обсязі.

Для реалізації серверної частини використовувались такі технології як Python, Django REST Framework, MySQL, Django ORM. Також, були використані такі технології як TensorFlow, Keras, ImageAI, numpy, scipy для побудови алгоритмів машинного навчання. Вони були реалізовані як частина серверної частини веб-системи.

Буди виконані такі кроки:

1. Додавання web-камери
2. Додавання набору даних для визначення об’єкту на зображенні
3. Тренування моделі для визначення зображень.
4. Захват об’єкту з вхідного зображення.
5. Виявлення пошкоджень на об’єкті.
6. Обробка зібраних даних.
7. Побудова графічного представлення результатів.

Для більш чіткого представлення алгоритму сегментації та кроків виконання з ним можна ознайомитися на рисунку 5.2.

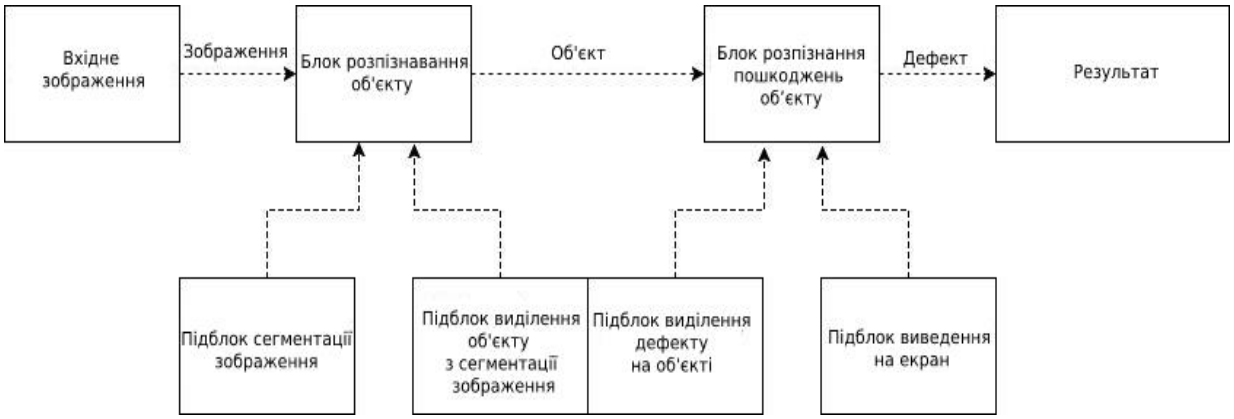


Рисунок 5.2 –Етапи роботи програми для розпізнавання пошкодження об’єкту

Також продемонстрована діаграма прецедентів, на якій зображені основні дії, що може виконувати користувач, рисунок 5.3.

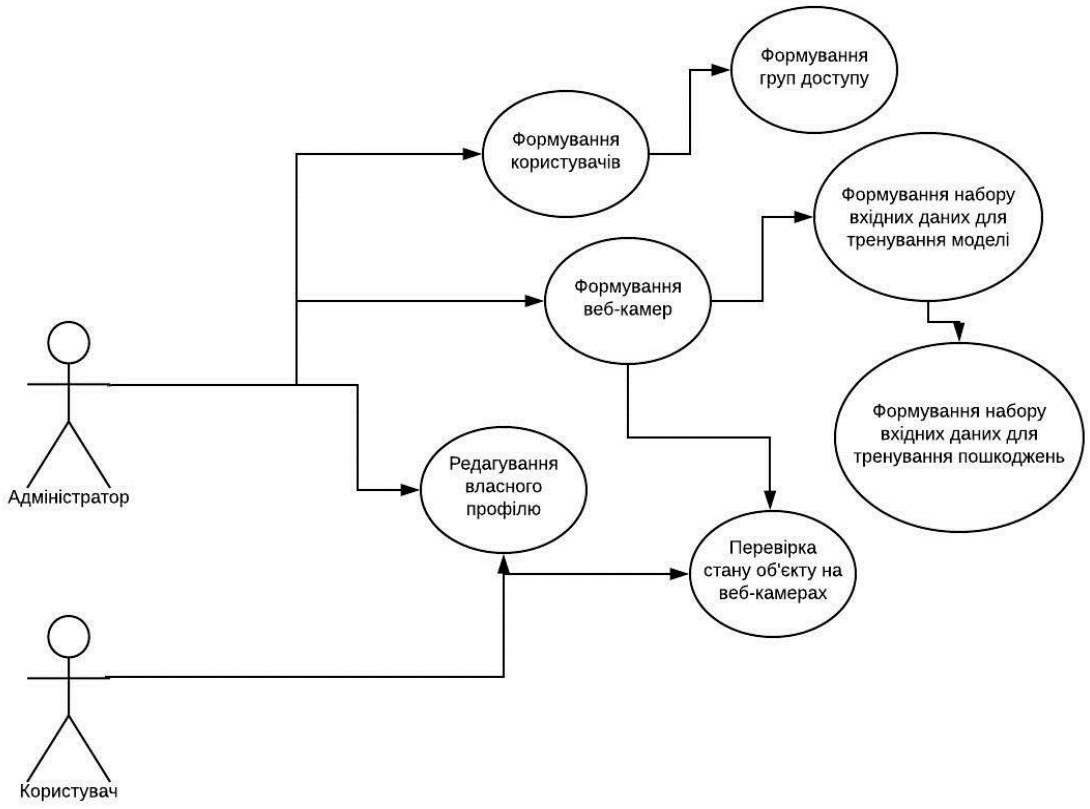


Рисунок 5.3–Діаграма прецедентів программного продукту.

5.3 Керівництво користувача

Для того, щоб скористатись функціоналом системи моніторингу користувач повинен увійти в систему, рисунок 5.4.

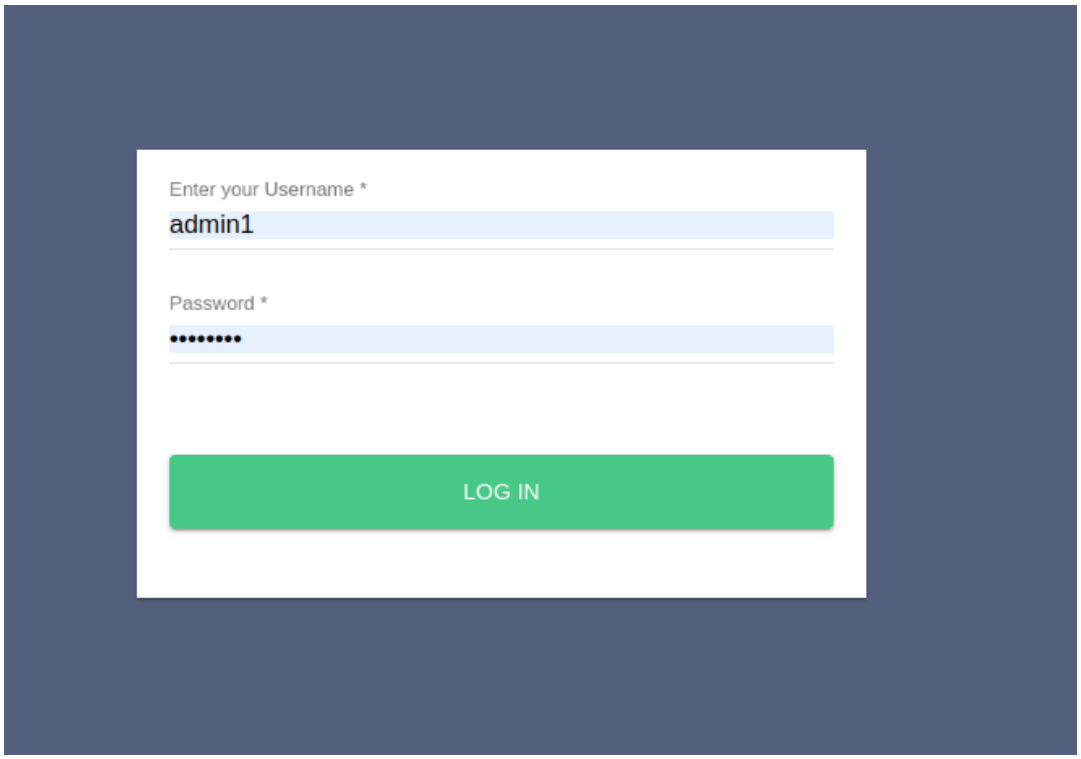


Рисунок 5.4–Форма входу у систему існуючого користувача

Після входження в систему користувач потрапляє на контрольну панель програми де може додавати веб-камери та набори даних, рисунок 5.5

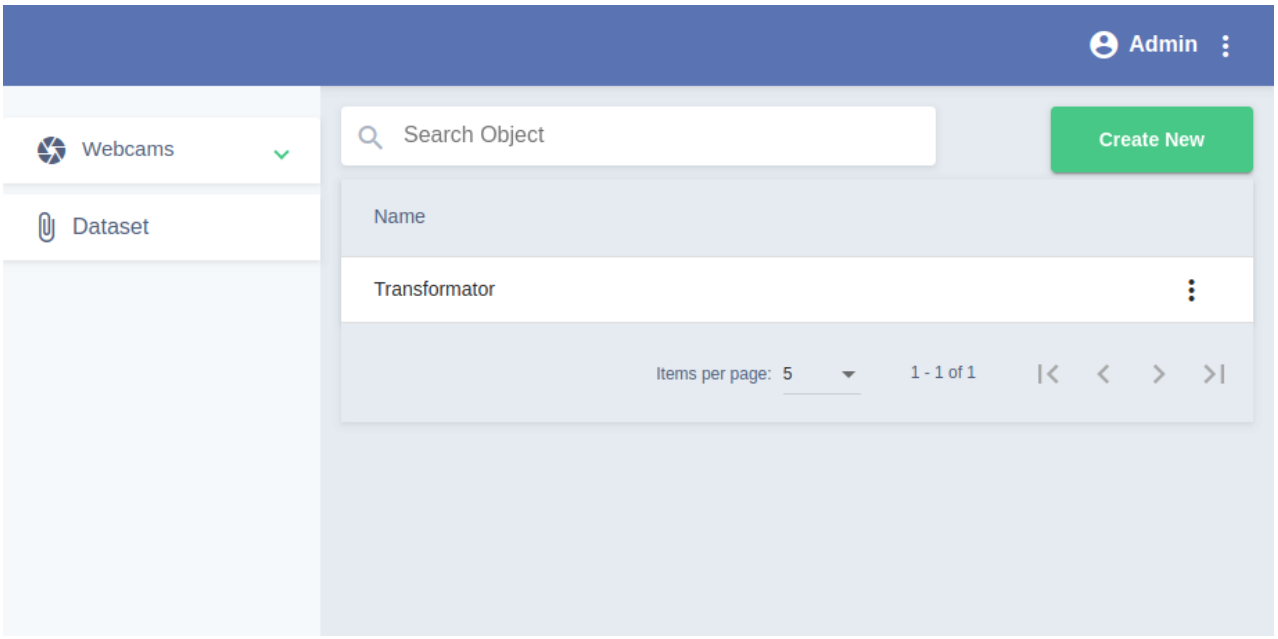


Рисунок 5.5–Головна панель web-системи

Також, користувач має право редагувати власний профіль перейшовши у верхнє праве випадаюче меню на спеціальній сторінці як зображено на рисунку 5.6.

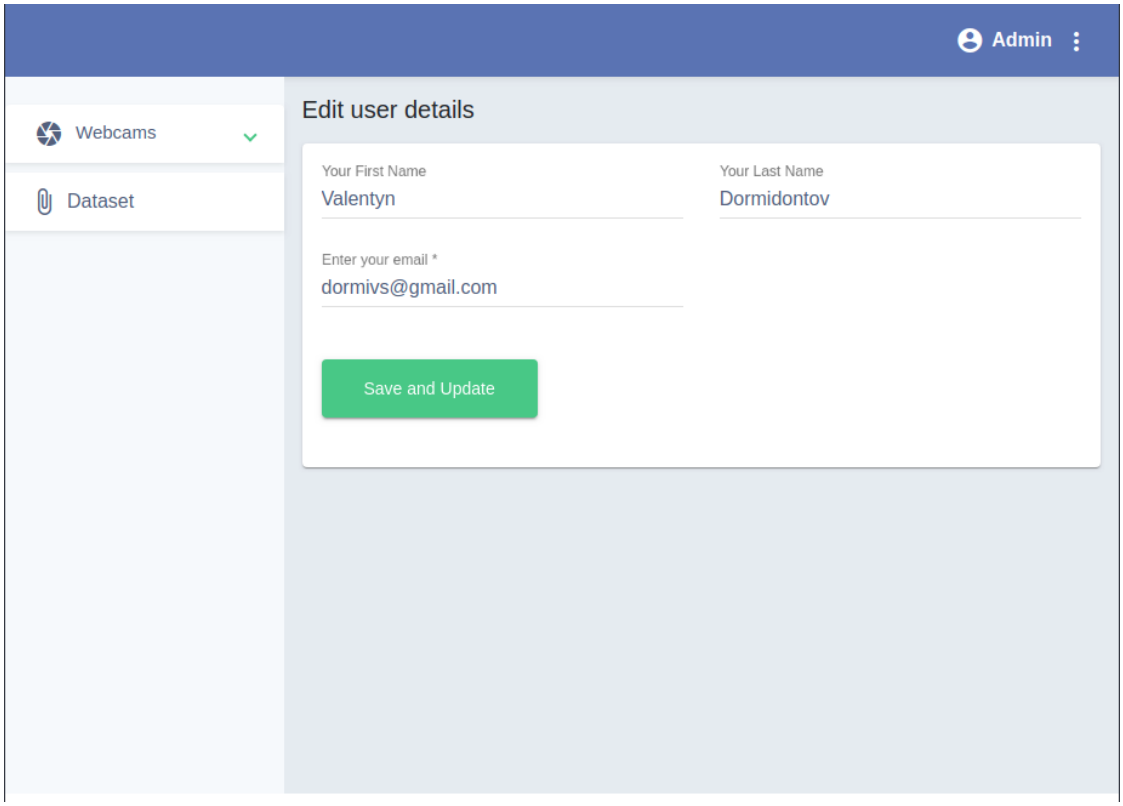


Рисунок 5.6–Редагування власного профілю

Якщо користувач є адміністратором, йому доступні також сторінки управління користувачами та групами, що зображені на рисунках 5.7 та 5.8.

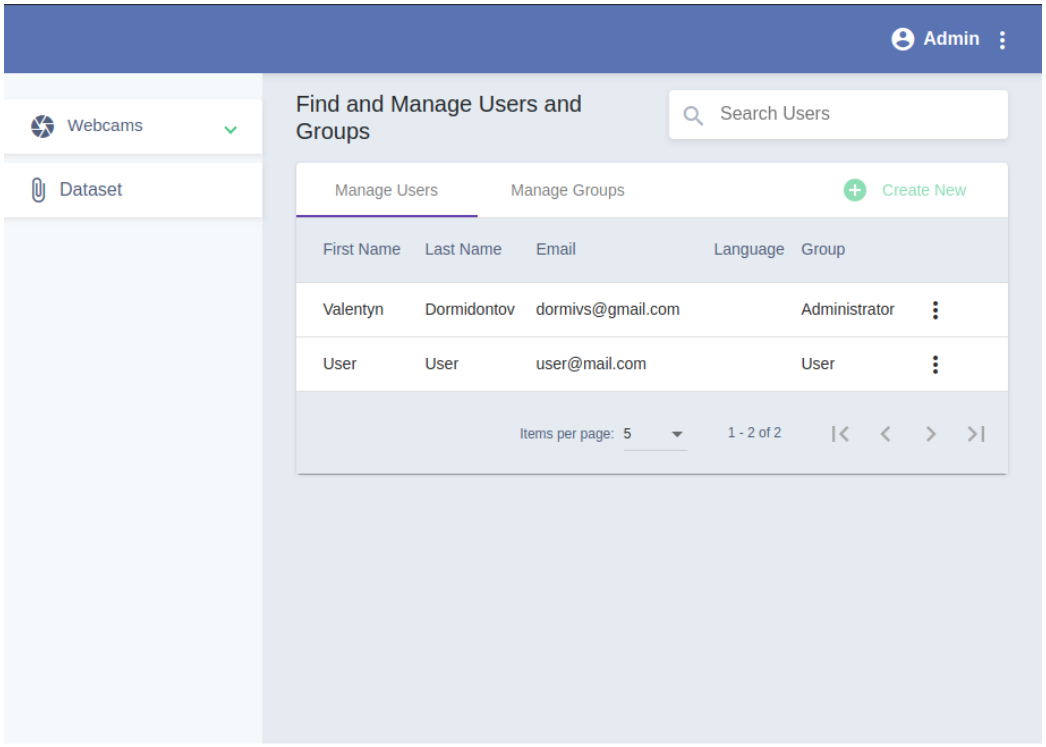


Рисунок 5.7–Сторінка управління користувачами

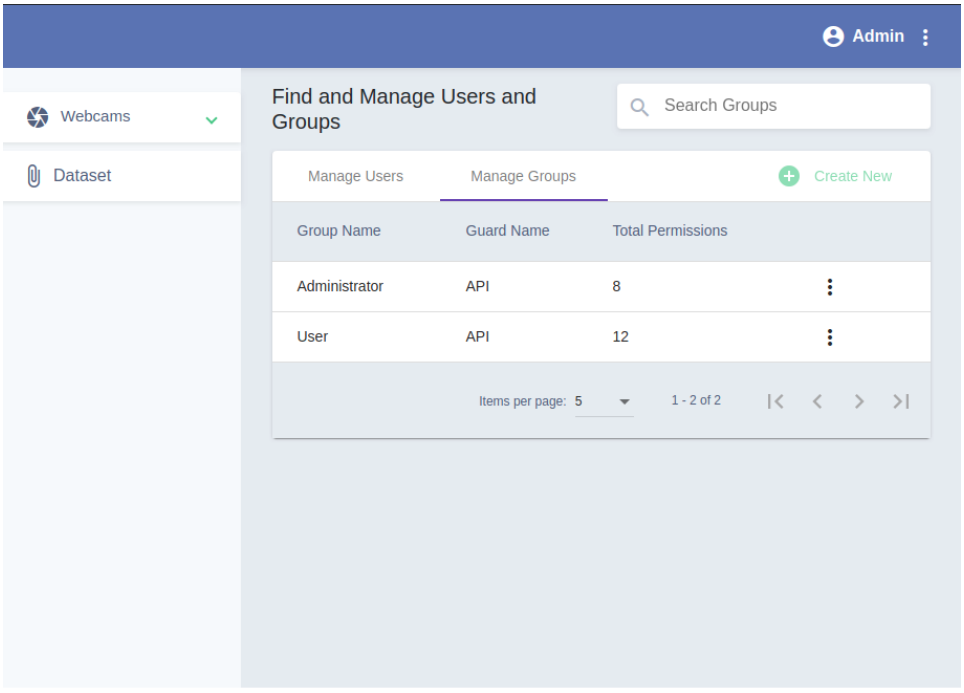
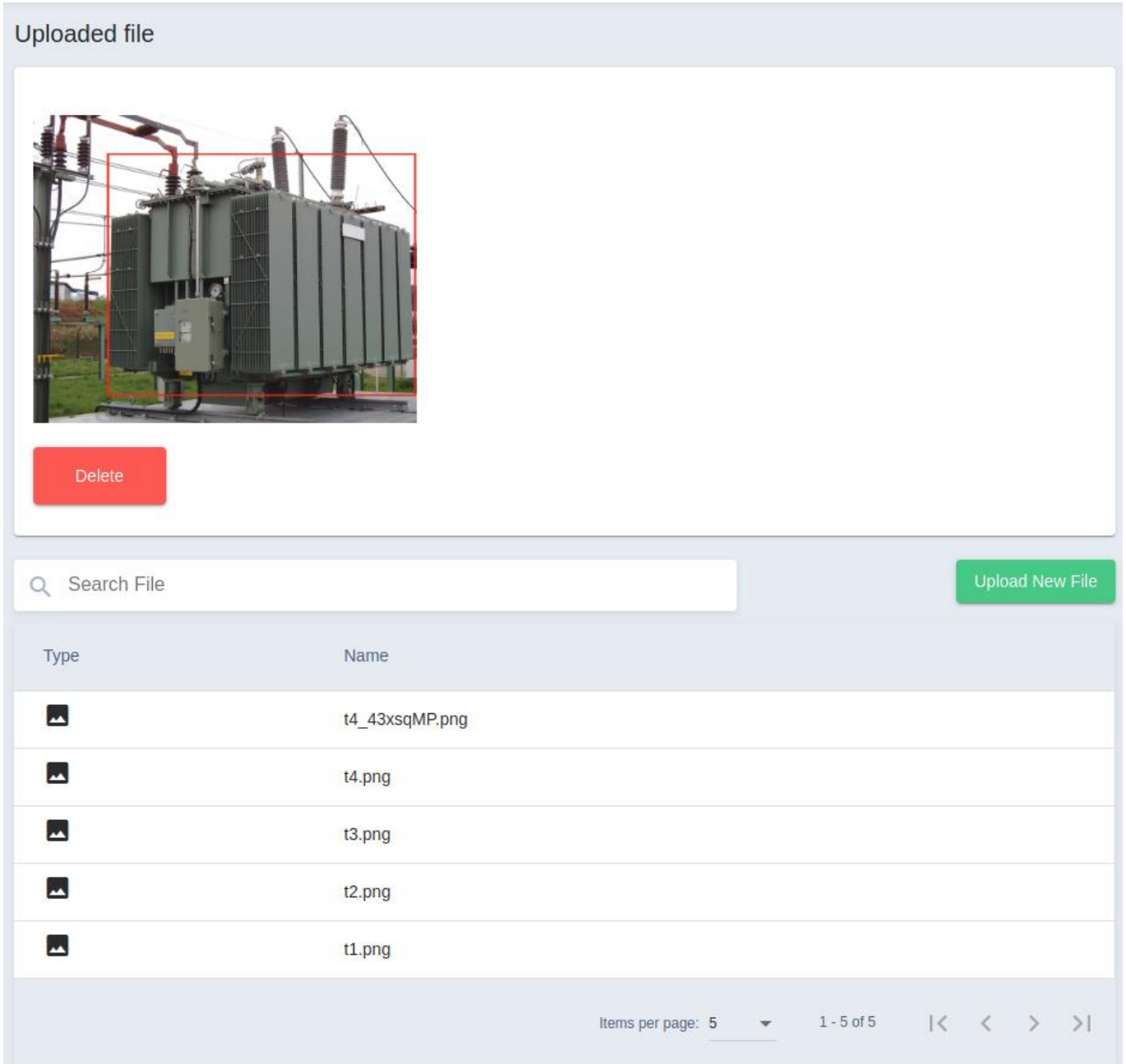


Рисунок 5.8–Сторінка управління групами

Адміністратор може створювати нових користувачів, групи та налагоджувати доступи до тих чи інших областей програми, а саме web-камер, наборів даних, користувачів, тощо.

Для виконання моніторингу за об’єктом необхідно заповнити набір даних для ідентифікування об’єкта, а саме завантажити декілька зображень об’єкту обведених



в рамку як показано на рисунку 5.9.

Рисунок 5.9–Складання набору даних

Далі, надати системі ресурс даних з web-камери. Система розрахована на програмне під’єднання камери до серверу та зберігання фото з камери в виділеній папці у файловій медіа-структурі системи, а також вказати назву, опис та обраний набір даних як показано на рисунку 5.10.

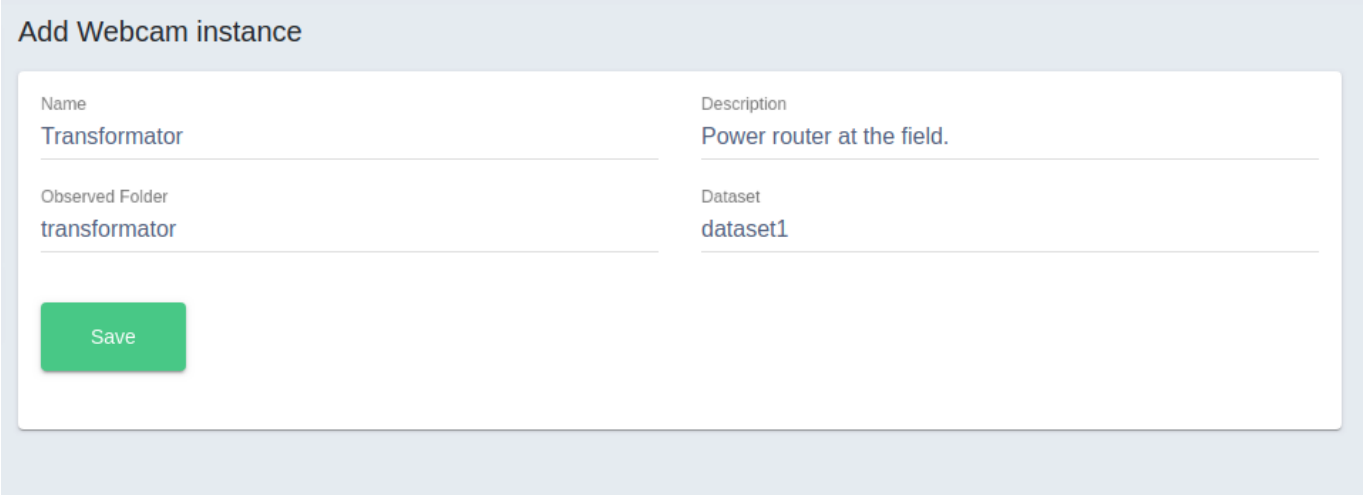


Рисунок 5.10–Додання камери

Після додавання одразу розпочнеться тренування моделі. Результат доданої камери можна переглянути на рисунку 5.11.

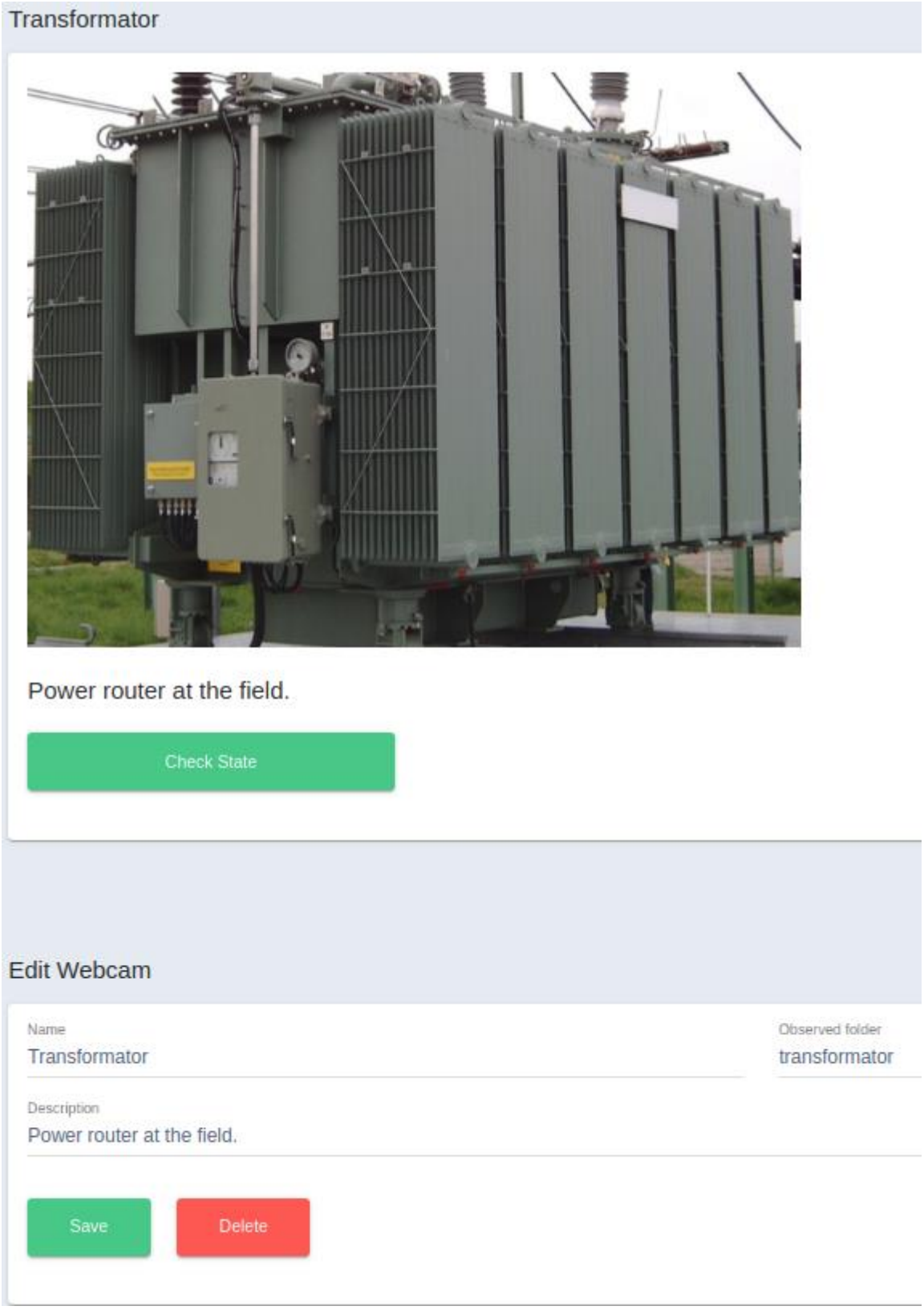


Рисунок 5.11–Додана камера

Для перевірки стану споглядаємого об’єкту користувач повинен натиснути кнопку “Check State”. Програма запросить з вказаної папки останнє за датою записане зображення та розпочне сканування об’єкту. В результаті цього користувачу буде виведений результат як на рисунку 5.12.

Рисунок 5.12–Результат роботи виявлення дефекту.

Користувачеві виводиться проаналізоване та оброблене зображення з маркованим дефектом.

5.4 Висновки до розділу

У розділі було розкрито структуру програмного продукту, методи, що були використані при розробці та керівництво користувача, де були описані весь функціонал.

Головні сторінки - це сторінка завантаження тренувальних даних на сервер та камер спостереженя задля моніторингу об’єкту, його точній ідентифікації та виявленню пошкоджень.

Було сформоване детальне керівництво користувача для полегшення у користуванні програмним продуктом. Керівництво користувача містить необхідні інструкції для швидкого ознайомлення із усім функціоналом системи. Також присутні screenshot’и кожної сторінки web-системи.

ВИСНОВКИ

В ході виконання дипломної роботи був виконаний аналіз проблем сегментації та обробки зображень використовуючи останні алгоритми та технології.

В ході виконання роботи виконено ознайомлення за такими бібліотеками як Keras, Tensorflow.

Було розроблено програмне забезпечення у вигляді web-системи по обробці динамічних зображень на прикладі фото та розпізнавання на ньому об`єктів. Також виконана реалізація аналізу об’єктну на пошкодження за набором даних та їх маркування з виводом на екран для розгляду користувача.

Реалізовано функціонал обробки результатів сегментації динамічного набору зображень на прикладі промислових об’єктів.

Реалізовано виявлення об’єкту на зображенні.

Виведено результат обробки зображення об’єкту з маркуванням його пошкоджень для подальшого аналізу користувачем.

Система спроектована з урахуванням можливості розширення функціоналу в майбутньому.

Виконано усі задачі, необхідні для досягнення мети роботи, а саме обробка та сегментація зображень задля моніторингу довкілля.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Novoseltsev IV Hybrid Neural Networks for Pattern Recognition / IV Novoseltsev., N.G. Aksak -M: Information Processing Systems, 2007. - Vip. 3.

2. Комп`ютерний зір. [Електронний ресурс]/Режим доступу:https://www.amazon.com/Computer-Vision-Modern-Approach-2nd/dp/013608592X

3. Pratt W. Digital image processing / W. Pratt - M .: Mir, 1982.

4. Обертоны и спектр [Електронний ресурс] – Режим доступу до ресурсу: http://asmir.info/lib/acoustics4.htm.

5. Цифрова обробка зображень. [Електронний ресурс]/Режим доступу:http://padabum.com/d.php?id=19283

6. Soifer V.A. Methods of computer image processing / V.A. Soifer - Moscow: FIZMATLIT, 2003.

7. Теоретичні основи цифрової обробки. [Електронний ресурс]/Режим доступу: https://www.twirpx.com/file/1412600/

8. Gonzalez R. Digital image processing / R. Gonzalez, R. Woods - M .: Technosphere, 2012.

9. Anisimov B. V. Recognition and digital image processing // BV Anisimov, VD Kurganov, VK Zlobin - M .: Vyssh., 1983.

10. Shapiro L. Computer vision // L. Shapiro, D. Stockmann - M .: Binom. Laboratory of knowledge, 2006.

11. Yu. V. Vizilter. Processing and analysis of images in computer vision problems. // Yu. V.Visilter, S. Yu. Zheltov, - Moscow: Fizmatkniga, 2010.

12. Zhuravlev Yu.I. Image recognition and image recognition // Yu. I. Zhuravlev, I.B. Gurevich - Moscow: Science, 1989.

13. Janet B. Digital image processing / B. Yane - M .: Technosphere, 2007.

ДОДАТОК А

Web-система моніторингу за набором зображень

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ42136_17Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ22112_17Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ22112_17Б 12-1	Vision	Основний компонент
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ22112_17Б 12-2	Users	Компонент керування користувачів

ДОДАТОК Б

Web-система обробки та сегментації динамічних зображень

Текст програми

УКР.НТУУ "КПІ". ТМ22112_17Б 12-1

Аркушів 11

2019

users.models.py

```
from django.contrib.auth.models import AbstractUser, Group as DjangoGroup
from django.db import models


class CustomUser(AbstractUser):
    name = models.CharField(blank=True, max_length=255)
    lang = models.CharField(blank=True, max_length=255)
    display_name = models.CharField(blank=True, max_length=255)
    email = models.EmailField(unique=True)
    image = models.ImageField(upload_to='images/upload', blank=True)
    mason_group = models.ForeignKey(
        DjangoGroup,
        null=True,
        blank=True,
        on_delete=models.SET_NULL,

        help_text='User group used for access separation between chat `Flow`s',
    )

    def __str__(self):
        return self.email
```

user.serializers.py

```
from rest_framework import serializers
from django.contrib.auth.models import Group, Permission
from django.contrib.auth.hashers import make_password
from . import models


class PermissionSerializer(serializers.ModelSerializer):
```

```
    class Meta:
        model = Permission
        lookup_field = 'id'
        fields = ('id', 'name', 'codename')


class PermissionPrivateSerializer(serializers.ModelSerializer):
    class Meta:
        model = Permission
        fields = ('id', 'name')


class GroupSerializer(serializers.ModelSerializer):
    class Meta:
        model = Group
        lookup_field = 'id'
        fields = ('url', 'id', 'name', 'permissions')


class GroupListSerializer(serializers.HyperlinkedModelSerializer):
    permissions = PermissionPrivateSerializer(many=True)

    class Meta:
        model = Group
        fields = ('url', 'id', 'name', 'permissions')


class GroupPrivateSerializer(serializers.ModelSerializer):
    class Meta:
        model = Group
        fields = ('name', 'id', )


class UserDetailsSerializer(serializers.ModelSerializer):
```

```
    class Meta:
        model = models.CustomUser
        fields = ('pk', 'first_name', 'last_name', 'email', 'display_name', 'username', 'image', 'lang', 'groups',
'password', )


class UserListSerializer(serializers.HyperlinkedModelSerializer):
    groups = GroupPrivateSerializer(many=True)

    class Meta:
        model = models.CustomUser
        fields = ('pk', 'first_name', 'last_name', 'email', 'display_name', 'username', 'image', 'lang', 'groups',
'password', )


class UserSerializer(serializers.ModelSerializer):

    def validate_password(self, value: str) -> str:
        return make_password(value)

    class Meta:
        model = models.CustomUser
        fields = ('pk', 'first_name', 'last_name', 'email', 'display_name', 'username', 'image', 'lang', 'groups',
'password', )
```

users.views.py

```
from rest_framework import viewsets
from django.contrib.auth.models import Group, Permission
from django.db.transaction import atomic
from rest_framework.permissions import IsAdminUser, IsAuthenticated
from .models import CustomUser
from rest_framework.response import Response
```

```
from .serializers import GroupSerializer, UserSerializer, PermissionSerializer, UserListSerializer, GroupListSerializer

# granting global permissions to our CRUDs
# permission_rule = [IsAdminUser]


class UserViewSet(viewsets.ModelViewSet):
    """
    API endpoint that allows users to be viewed or edited.
    """
    queryset = CustomUser.objects.all()
    serializer_class = UserSerializer
    permission_classes = (IsAuthenticated, )

    def list(self, request):
        queryset = CustomUser.objects.all()
        serializer = UserListSerializer(queryset, many=True)
        return Response(serializer.data)
    # permission_classes = permission_rule

    def _reset_chat_authorization(self):
        # User Groups can be changed
        # (needs reset authorization timestamp)
        user = self.get_object()
        user.chat_logs.authorized_now().update(authorized_at=None)

    @atomic
    def update(self, *args, **kwargs):
        self._reset_chat_authorization()
        return super().update(*args, **kwargs)

    @atomic
    def partial_update(self, *args, **kwargs):
```

```
        self._reset_chat_authorization()
        return super().partial_update(*args, **kwargs)

    def destroy(self, request, pk=None, permission_classes=(IsAdminUser,)):
        pass


class GroupViewSet(viewsets.ModelViewSet):
    """
    API endpoint that allows groups to be viewed or edited.
    """
    queryset = Group.objects.all()
    serializer_class = GroupSerializer
    permission_classes = (IsAuthenticated, )

    def list(self, request):
        queryset = Group.objects.all()
        serializer_context = {
            'request': request,
        }
        serializer = GroupListSerializer(queryset, context=serializer_context, many=True)
        return Response(serializer.data)
    # permission_classes = permission_rule


class PermissionViewSet(viewsets.ModelViewSet):
    """
    API endpoint that allows permissions to be viewed or edited.
    """
    queryset = Permission.objects.all()
    serializer_class = PermissionSerializer
    # permission_classes = permission_rule
```

vision.models.py

```
from django.db import models


# Create your models here.

class Dataset(models.Model):
    id = models.AutoField(primary_key=True, help_text="Dataset ID")
    name = models.CharField(blank=False, unique=True, max_length=255, help_text="Internal name for Dataset.")
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)


class WebcamObject(models.Model):
    id = models.AutoField(primary_key=True, help_text="Webcam ID")
    name = models.CharField(blank=False, unique=True, max_length=255, help_text="Internal name for Webcam.")
    description = models.CharField(blank=False, max_length=255, help_text="Short description")
    folder = models.CharField(blank=False, default=3306, max_length=255, help_text="Folder where incoming data will be stored")
    image = models.ImageField(blank=True, upload_to='images/upload')
    dataset = models.ForeignKey(Dataset, on_delete=models.PROTECT, help_text="Dataset ID", blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

vision.serializers.py

```
from rest_framework import serializers
from . import models

class DatasetSerializer(serializers.ModelSerializer):
```

```
    class Meta:
        model = models.Dataset
        lookup_field = 'id'
        fields = (
            'id',
            'name',
            'created_at',
            'updated_at',
        )


class WebcamObjectSerializer(serializers.ModelSerializer):
    class Meta:
        model = models.WebcamObject
        lookup_field = 'id'
        fields = (
            'id',
            'name',
            'description',
            'folder',
            'image',
            'dataset',
            'created_at',
            'updated_at',
        )
```

vision.views.py

```
from django.shortcuts import render
from rest_framework import viewsets, status
from rest_framework.decorators import action
from rest_framework.permissions import IsAdminUser, IsAuthenticated
from . import models, serializers
from .utils import getobj
```

```
from rest_framework.response import Response

class DatasetViewSet(viewsets.ModelViewSet):
    """
    API endpoint that allows External Databases settings to be viewed or edited.
    """
    queryset = models.Dataset.objects.all()
    serializer_class = serializers.DatasetSerializer
    permission_classes = (IsAuthenticated, )


class WebcamObjectViewSet(viewsets.ModelViewSet):
    """
    API endpoint that allows External Databases settings to be viewed or edited.
    """
    queryset = models.WebcamObject.objects.all()
    serializer_class = serializers.WebcamObjectSerializer
    permission_classes = (IsAuthenticated, )

    @action(detail=True, methods=['get'])
    def test_state(self, request, pk=None):
        webcam = self.get_object()

        return Response({'detail': "Test finished", "img": getobj(webcam)})
```

vision.utils.py

```
import tensorflow as tf
import sys
import os
import urllib
```

```
# Disable tensorflow compilation warnings
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
import tensorflow as tf

def getobj(image_path):
    # Read the image_data
    image_data = tf.gfile.FastGFile(image_path, 'rb').read()
    print(image_path)

    # Loads label file, strips off carriage return
    label_lines = [line.rstrip() for line
                   in tf.gfile.GFile(r"./models/tf_files/retrained_labels.txt")]

    # Unpersists graph from file
    with tf.gfile.FastGFile(r"./models/tf_files/retrained_graph.pb", 'rb') as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f.read())
        _ = tf.import_graph_def(graph_def, name='')

    with tf.Session() as sess:
        # Feed the image_data as input to the graph and get first prediction
        softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

        predictions = sess.run(softmax_tensor, \
                {'DecodeJpeg/contents:0': image_data})

        # Sort to show labels of first prediction in order of confidence
        top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]


        for node_id in top_k:
            count = 1
            human_string = label_lines[node_id]
            score = predictions[0][node_id]
```

```
print(count)
count += 1
print('%s (score = %.5f)' % (human_string, score))
score = (round((score * 100), 2))
return human_string,score
```

ДОДАТОК В

WEB-система моніторингу за набором зображень

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ АПЕПС_ТМ22112_18Б 13-1

Аркушів 10

Київ 2018

АНОТАЦІЯ

Додаток містить опис web-системи по моніторингу довкілля за набором зображень, що виконує деякі із завдань, поставлених в розділі 1, а саме:

— Завантаження фото на сервер та його обробка і сегментація для визначення об`єкта;

— Ідентифікація пошкоджень на визначеному об’єкті;

— Реалізація зручного інтерфейсу для доступу до функціоналу додатку.

Додаток розроблений за допомогою мови програмування Python з використанням технології Django.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис основного компоненту web-системи по моніторингу довкілля за набором зображень, що виконує деякі із завдань, поставлених в розділі 1. У додатку Б міститься програмний код компоненту.

Система для роботи потребує бразузер, що підтримує JavaScript та Cookies.

Додаток розроблений за допомогою мови програмування Python з використанням технології Django.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Додаток надає функціонал для роботи зі моніторингом пошкоджень по обробленим даним із обробленого фото, а саме:

— завантаження фото для обробки на сервер;

— ідентифікація об’єкту;

— ідентифікація пошкоджень;

— вивід модифікованого зображення засобами інтерфейсу користувача системи

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою, додаток складається з трьох головних модулів:

— клієнт (frontend)

— сервер (backend)

— сервер баз даних

Клієнт містить в собі весь візуальний інтерфейс (UI) для взаємодії з користувачем системи.

Сервер містить в собі класи та інтерфейси, що використовуються для моделювання сутностей, якими оперує додаток.

Сервер баз даних містить класи з методами для створення, видалення та оновлення полів у базі.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання додатку користувач повинен мати персональний комп’ютер або мобільний пристрій з браузером, що підтримує JavaScript, а також підключення до мережі Інтернет.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програма не потребує інсталяції і доступна на сайті за посиланням http://localhost:4200

ВХІДНІ ДАНІ

Вхідна інформація для додатку:

1) дані для створення індивідуального облікового запису;
2) дані для обробки.

ВИХІДНІ ДАНІ

Вихідна інформація додатку:

1) Оброблені зображення з ідентифікованим об’єктом та маркованими дефектами